# Assignment-2/ Project

## Student information

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Roll No | Role in the project | Remarks |
| Sarthak Patel | R2142201919 | Preprocessing and Documentation |  |
| Mrigrendra | R214220715 | Model Building |  |
| Manan Gupta | R214220659 | Web Deployment |  |
| Anurag Singh | R214220205 | Web Deployment |  |

## Project title

Sentiment analysis of Youtube comments.  
  
Basically in this project we have build a web portal that performs sentiment analysis on YouTube video comments and that is valuable tool for understanding the overall sentiment surrounding a video.

## Dataset used

Link 🡪 <https://www.kaggle.com/competitions/sentiment-analysis-on-movie-reviews>

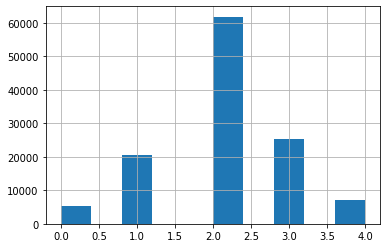
The Rotten Tomatoes dataset is divided into tab-separated files that include phrases. For benchmarking reasons, the train/test split has been kept, but the sequence of the phrases has been changed. The Stanford parser has broken down each sentence into a variety of phrases. Every sentence has a Phrase-Id. A Sentence-Id identifies each sentence. Recurring phrases (such short, popular words) are only recorded once in the data.

The phrases and their corresponding sentiment labels may be found in train.csv. To make it easier for you to keep track of which phrases are part of a single sentence, we have also included a Sentence-Id.

There are just phrases in test.csv. Each word has to have a sentiment label applied to it.

The sentiment labels are:

0 - negative  
1 - somewhat negative  
2 - neutral  
3 - somewhat positive  
4 - positive



## Methods used We have used Ensemble method .

Preprocessing: Preprocess the extracted comments by removing any irrelevant information, such as URLs, emojis, or special characters. This step helps to clean the data and make it suitable for sentiment analysis.  
  
Stemming: Stemming is a process that reduces words to their root or base form. It helps to handle different variations of words and reduces the dimensionality of the feature space.

Stopword Removal: Stopwords are commonly occurring words in a language that do not carry significant meaning or contribute much to the overall sentiment analysis task. Examples of stopwords include "the," "is," "and," "a," and so on. These words can be safely removed to reduce noise and improve computational efficiency.

Converting Text to Lowercase: Converting all text to lowercase is a common step in text preprocessing. It helps to ensure that words are treated consistently, irrespective of their casing. By converting text to lowercase, you make the analysis case-insensitive and avoid treating the same word as different simply because of capitalization.  
  
CountVectorizer: CountVectorizer is a simple and widely used technique for converting a collection of text documents into a matrix of token counts. It operates by tokenizing the text and counting the occurrences of each token in the document.  
  
TF-IDF (Term Frequency-Inverse Document Frequency): TF-IDF is a numerical statistic that reflects the importance of a word in a document collection or corpus. It combines two measures: term frequency (TF) and inverse document frequency (IDF).  
  
Base Models: We have selected a set of base models for sentiment analysis. These models can be different algorithms or variations of the same algorithm trained with different parameters or feature representations. For example, you might use a Random Forest model, a Support Vector Machine (SVM) model, Logistic regression and a Multinomial Naive Bayes (MNB) model.

Training: Train each base model using a labeled sentiment analysis dataset. The dataset should include comments and their corresponding sentiment labels.

Prediction: Feed the comments from the YouTube video into each base model to obtain individual sentiment predictions. Each base model will assign a sentiment label (positive, negative, or neutral) to each comment.

Voting: Use a voting mechanism to combine the predictions of the base models. There are different types of voting, such as majority voting and weighted voting.

Majority Voting: In this approach, each base model gets one vote, and the sentiment label with the most votes becomes the final prediction. If there is a tie, you can choose the sentiment label randomly or consider using a predefined priority order.

Weighted Voting: Assign a weight to each base model's prediction based on its performance or confidence level. The weights can be manually assigned or determined automatically using techniques like cross-validation or model evaluation metrics. The sentiment label with the highest weighted sum becomes the final prediction.

Aggregate Sentiment: Calculate the overall sentiment percentage based on the combined predictions. You can assign scores to each sentiment label (e.g., +1 for positive, -1 for negative, 0 for neutral) and sum up the scores. Finally, calculate the percentage based on the total number of comments.

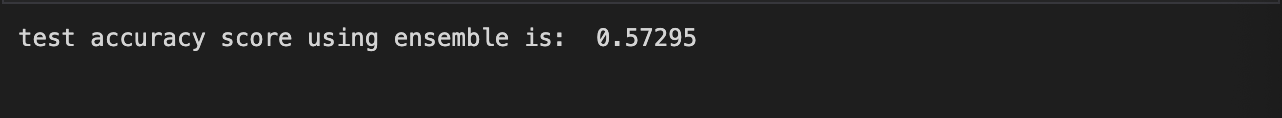
Output and Visualization: Display the calculated sentiment percentage and any additional visualizations or insights on the web portal.

Remember to evaluate the performance of each base model individually to ensure they provide meaningful contributions to the ensemble. Also, consider the potential class imbalance in the dataset and adjust the voting mechanism accordingly, if needed.

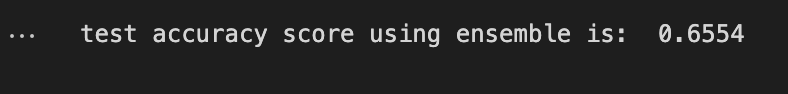
By combining the predictions of multiple models through voting, the ensemble approach can help mitigate biases and improve the overall accuracy and robustness of sentiment analysis.

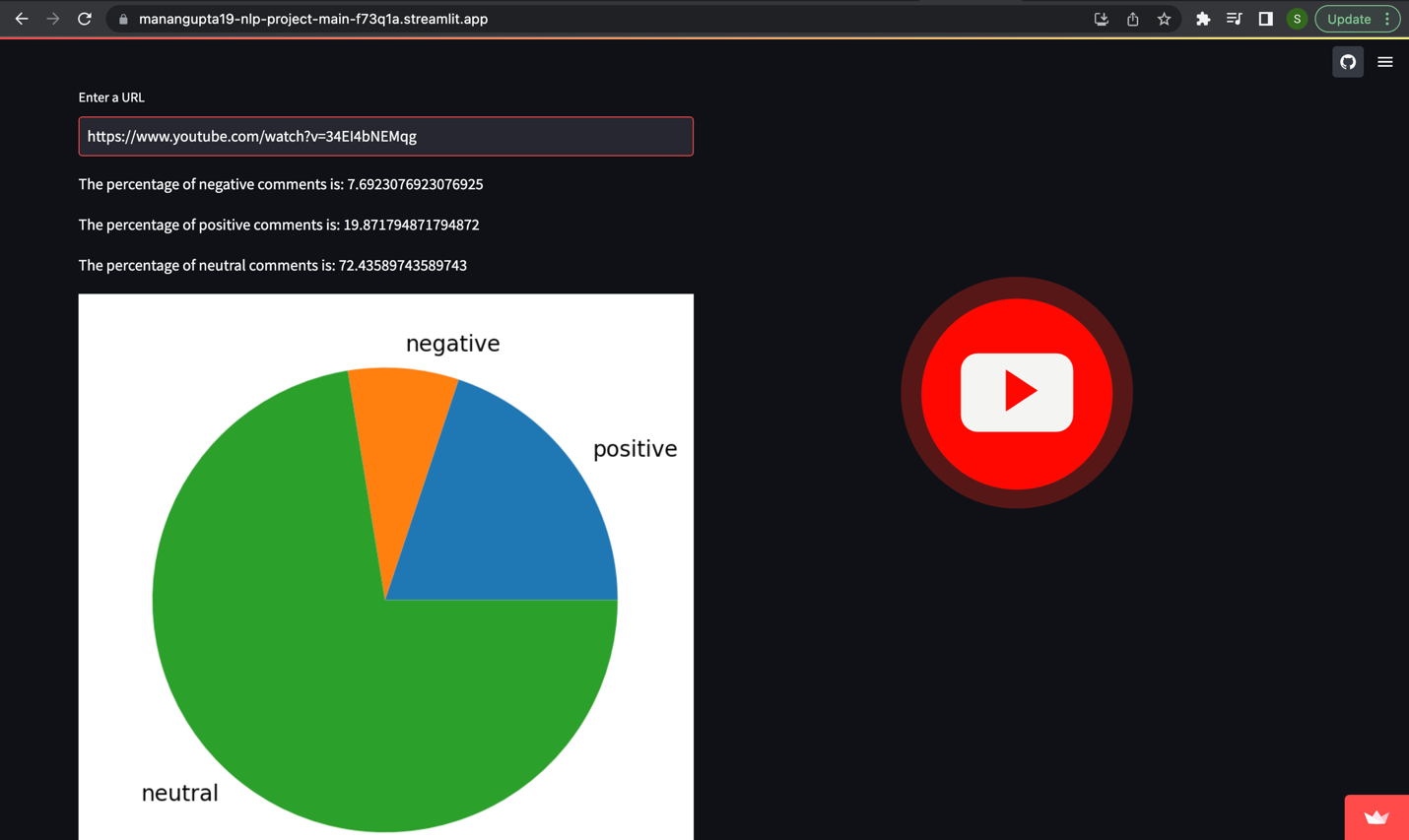
## Results

🡪Before Presentation   
We have taken 5 class (0 – negative, 1 - somewhat negative, 2 – neutral, 3 - somewhat positive, 4 – positive )



🡪After Presentation   
  
We have taken 3 class (0 – negative, 1 – neutral, 2 – positive )





Link of web portal 🡪 <https://manangupta19-nlp-project-main-f73q1a.streamlit.app/>

Challenges or learning throughout project development (3–10 points)  
  
During the development of your sentiment analysis project, we encounter various challenges and learning opportunities.  
  
🡪 Data Collection and Quality: Gathering a labeled dataset for sentiment analysis can be challenging. That’s we have taken a movie review dataset.  
🡪 Model Selection and Evaluation: Choosing the right base models for your ensemble can be challenging. It requires understanding the strengths and weaknesses of different algorithms and architectures. Evaluating the performance of individual models and the ensemble is important. We learn about model evaluation metrics, cross-validation, and hyperparameter tuning.  
🡪 Handling Class Imbalance: Sentiment analysis datasets often suffer from class imbalance, where one sentiment class dominates the others. This can lead to biased predictions. Learning to handle class imbalance through techniques like oversampling, undersampling, or using class weights can help improve the performance of your sentiment analysis system.  
🡪 Deployment and Scalability: Deploying your sentiment analysis system as a web portal involves considerations such as server infrastructure, scalability, security, and user experience. Learning about web development frameworks, cloud services, and performance optimization can be valuable.